Run BASIC – A Breakthrough Web Application Server

"Web programming for people who really like to program!"
http://www.runbasic.com

Carl Gundel, carlg@libertybasic.com

Run BASIC allows you to apply your desktop programming skills to create web applications without struggling with the overhead of typical web programming frameworks.

Unlike other web programming technologies, with Run BASIC there is no state to manage, you don't need to GET or POST anything, and you never even have to think about the Common Gateway Interface. You can easily deliver interesting and rich applications that are difficult to deliver with other tools. Run BASIC hides the web technology so you can just code.

Honoring a Tradition of Simplicity

Run BASIC is a version of the BASIC programming language. BASIC was designed from the start to be easy for the non-computer scientist to use. A small language with simple syntax and English-like commands, BASIC is easily learned. Modern implementations of BASIC also add important features such as support for structured programming and scoped subroutines. Inspired by the approachability of BASIC we decided to create a web programming system with those important qualities.

When BASIC was at the peak of its popularity in the 1980's most people who used it were users of home computers. These were usually 8-bit microcomputers that plugged into a television set. With the flick of a power switch the user would almost instantly see something like this:

```
SuperDuper BASIC v2.3
15485 Bytes Free
Ready
```

So the user was able to immediately begin typing commands and programming the machine. Instead of booting into Windows the computer booted into BASIC which was the operating system of the machine.

Run BASIC is similar in its conception. Everything you need to write web programs comes built right in . Just start it up and you're ready to create something useful. People familiar with web programming aren't prepared for its simplicity because they think that CGI programming is easy. People who've never written a web program before will quickly take it for granted that all web programming is so simple.

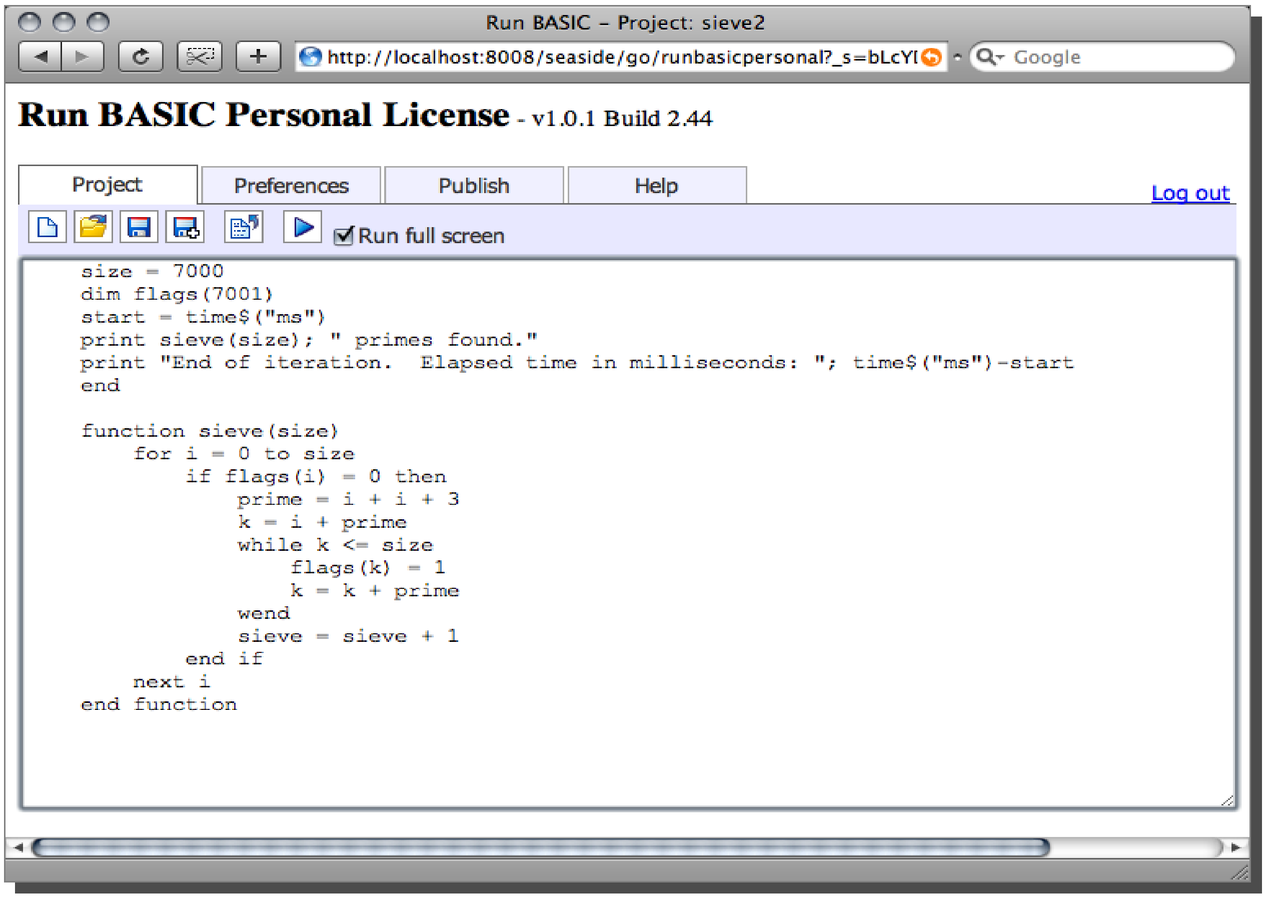
The Foundation

Some web programming languages are modules that you load into a web server like Apache. The trouble with this approach is that you need to know about web server technology and programming too. Actually there's more to it than this because you also need to learn about session management and about preserving state between requests. If you want to draw graphics or work with a database you have to load separate modules.

Because we believe that most people would prefer to just get down to programming and forget about all the complications of web technology, Run BASIC is built on top of a web server and it hides all the complexity. It includes everything and provides a friendly face.

Getting Started

Run BASIC has a single installer. Once you have it running a single button launches your web browser and connects it to the Run BASIC server. This becomes your programming interface. You write code and test it right away in the browser. You can type a little code and click the Run button to see it run right away.


Run BASIC – Project: sieve2
http://localhost:8008/seaside/go/runbasicpersonal?_s=bLcYI
Google
Run BASIC Personal License - v1.0.1 Build 2.44
Project
Preferences
Publish
Help
Log out
Run full screen

```
size = 7000
dim flags(7001)
start = time$("ms")
print sieve(size); " primes found."
print "End of iteration.  Elapsed time in milliseconds: "; time$("ms")-start
end

function sieve(size)
    for i = 0 to size
        if flags(i) = 0 then
            prime = i + i + 3
            k = i + prime
            while k <= size
                flags(k) = 1
                k = k + prime
            wend
            sieve = sieve + 1
        end if
    next i
end function
```

In the tradition of the BASIC language, you can see all of your code in the one editor. This is a really important because most modern web tools force you to break your program code up into different pieces. Run BASIC also lets you put your code into separate files (or modules) if you want to but being able to see everything in one place is very handy for simple exploration or small projects.

Control Flow and State Management

One very special feature of Run BASIC is that you can write a simple interactive program in the classic terminal style. Creating this kind of software is a lot of work in most web programming systems. Here is a quick example:

```
'Multiplication table
[start]
  input "How many columns?"; cols
  input "How many rows?"; rows
  for x = 1 to cols
    for y = 1 to rows
      print x*y; " ";
    next y
    print
  next x
  input "Do it again (Y/N) "; yn$
  if yn$ = "y" or yn$ = "Y" then [start]
  end
```

BASIC programmers will recognize this as the sort of code they would write on their Commodore or TRS-80 computer, or using QBASIC. This simple example works exactly as you would expect it to, but web programmers don't get to write such a simple program with so little code. It must be broken up into a web page that asks for the columns and rows, a script file to generate another page with the table and to ask to do it again, and then another script file to process that request. The values for cols and rows also need to be passed from the first page to the script that generates the table.

Run BASIC does all this work for you automatically. There are fewer files and less code.

Automatically Generated HTML

Since Run BASIC is designed to let you get a lot done with a minimum of code it generates the necessary HTML code for you. Less code is the result. If you really do need some special HTML effect or if you are a Javascript coder you can embed as much HTML code as you like into your web application.

Another benefit of generated HTML is that Run BASIC will not forget to match up the opening and closing tags.

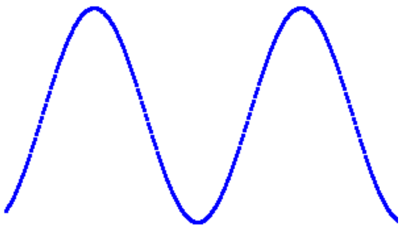
Built-in Facilities

Graphics

One thing that people find useful and fun is the ability to draw graphics, but this is not easy in web programming systems. Here is a very simple program that creates a graphic object and then uses it to plot a sine wave. The RENDER statement at the end simply instructs Run BASIC to 'PRINT' the graphic to the web page.

```
'sinewave.bas - This example draws a sine wave
graphic #sineGraph, 300, 200
#sineGraph size(3)
for x = 1 to 300
  #sineGraph color("blue")
  #sineGraph set(x, 85*sin((50+x)/25)+100)
next x
render #sineGraph
end
```

The result:



One of our users wrote a graphical hangman with no prior web programming experience.

Database

Many times it is handy to keep information in a database, and so we decided to integrate the excellent SQLite database engine. The special qualities about this particular database include that it does not require installation or trained administrator and there is no need to tweak any settings. It just works. This simplicity fits well with the spirit of Run BASIC.

Here are a few lines of code that show how to get some data out of a database and show it on a web page:

```
sqliteconnect #stuff, "stuff.db"
#stuff execute("select ID,USERNAME,DESCRIPTION,EMAIL from entries")
render #stuff
#stuff disconnect()
```

Here is the result:

```
ID USERNAME        DESCRIPTION              EMAIL
25 Carl Gundel     A Big Bad BUG            carlg@libertybasic.com
26 Domenick Demoro Don't forget about this ddemoro@dd.com
28 Red Fox         Heart attack bug         rfoxx@sanford.com
```

Widgets

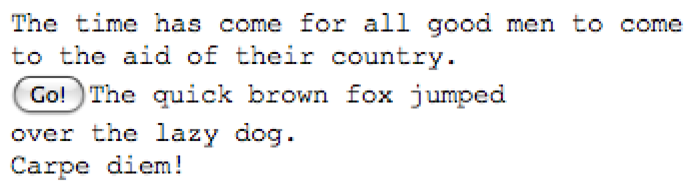
Creating a user interface with widgets like buttons, links, listboxes, etc. has been designed to be as simple as possible. The user PRINT statement to display text and numbers. This happens just like on a printed page from the top. Widgets are added into the flow of the page in the same way.

Here is an example:

```
  print "The time has come for all good men to come"
  print "to the aid of their country."
  button #go, "Go!", [makeItHappen]
  print "The quick brown fox jumped"
  print "over the lazy dog."
  wait

[makeItHappen]
  print "Carpe diem!"
  end
```

Here's what this looks like when you run it and click the button.



```
The time has come for all good men to come
to the aid of their country.
Go! The quick brown fox jumped
over the lazy dog.
Carpe diem!
```

The button invokes the [makeItHappen] handler. You can also specify a named subroutine if you prefer, like so:

```
  print "The time has come for all good men to come"
  print "to the aid of their country."
  button #go, "Go!", makeItHappen
  print "The quick brown fox jumped"
  print "over the lazy dog.'
  wait

sub makeItHappen key$
  print "Carpe diem!"
  print "You clicked on "; key$
end sub
```

The name #go gets passed in as a string to the key$ variable of the subroutine. Other widgets also use event handlers in the same way.

Components in Applications

Run BASIC provides a simple way to let programmers create components for their web applications by using the tried and true RUN statement in a novel way. Any program that is run can then be used as a component, which can optionally be rendered anywhere on a web page by using the same RENDER statement demonstrated above in the sections on graphics and database features.

For example here is a program which let's the user type in a color and click Go! to draw a sine wave of a desired color.

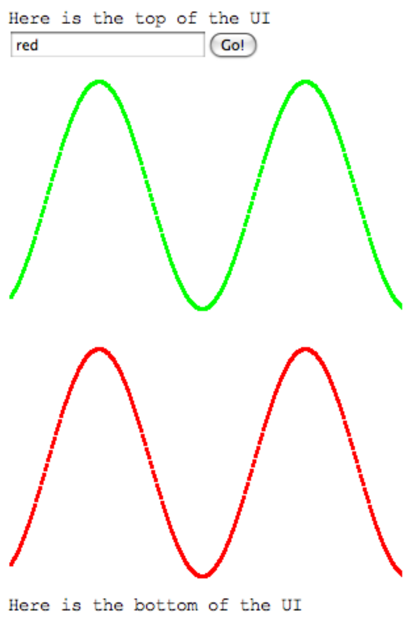
```
  'colorSine.bas
  textbox #tb, ""
  button #go, "Go!", [go]
  wait

[go]
  print
  color$ = #tb contents$()
  graphic #sineGraph, 300, 200
  #sineGraph size(3)
  for x = 1 to 300
    #sineGraph color(color$)
    #sineGraph set(x, 85*sin((50+x)/25)+100)
  next x
  render #sineGraph
  wait
```

And here is a very simple program that embeds the above program as a component onto the page.

```
  'embeddedSine.bas
  run "colorSine", #sine
  print "Here is the top of the UI"
  render #sine
  print
  print "Here is the bottom of the UI"
  wait
```

The embeddedSine.bas program runs all the way down to the WAIT statement. The colorSine.bas component remains interactive. The user can type in a color and click Go! as many times as desired and the structure of the embeddedSine.bas UI is maintained. For example here the user draws two sine waves:


```
Here is the top of the UI
```

```
Here is the bottom of the UI
```

Components do not need to be rendered onto the user interface. They can also be used as libraries of shared code or even as objects in an object oriented fashion.

Web 2.0 Features

Since Run BASIC is a web server it has commands for communicating with other Internet systems and since it is BASIC it makes sense for these commands to be really easy. You can programmatically:

- Fetch the contents of a web page from another server
- Submit information to another server and get a response
- Send email using SMTP

A built-in XML parser makes it easy to read information out of what you get back from other servers.

Support for Styling

Once you've got a program running and working just the way you like, Run BASIC's support for CSS and HTML makes it easy to pretty up your web application.

Run BASIC treats DIV tags as a first class structure by making a code block for it so the compiler can enforce good structure. Then you can apply CSS styles for those DIVs directly in the code.

Here is an example:

```
cssid #title, "{ font-family: Tahoma; font-size: 20pt; font-weight: bold }"
cssid #copyright, "{ font-size: 8pt }"
cssid #list, "{ font-family: Tahoma; background: #E8FFE8; padding: 4px }"
div title
  print "Blogging Example"
  div copyright
    print "Copyright 2007, Shoptalk Systems"
  end div
end div
div list
  for x = 1 to 5
      print word$("one two three four five", x)
  next x
end div
```

To Learn More

Visit our web site at http://www.runbasic.com and send email inquiries to carlg@libertybasic.com